Aplicaciones restful (3ra parte)

Documento de API

Teoría

Objetivo / Motivación

Plasmar en un solo lugar la lista con todos los recursos (entidades) representados en el servidor, con qué métodos (funcionalidades) cuenta cada recurso, qué se puede esperar de cada uno de ellos como resultado, qué características/restricciones tiene cada petición, qué tipo de errores puede devolver, y ante qué casos se producen los mismos, etc.

Componentes de la documentación

- Nombre y características de cada recurso
- Nombre, ruta y descripción de cada método
- Características de la petición y respuesta de cada método
- Posibles códigos de estado/error que puede devolver el servidor, y sus significados

Dado que no existe una única manera de confeccionar este documento, ni una manera “correcta”, elegimos la documentación de algunos sitios popularmente reconocidos y utilizados, para ejemplificar cada una de las partes del documento.

Nombre, descripción, y características de cada recurso

Como ya sabemos, un servidor rest representa un conjunto de recursos accesibles mediante peticiones. Es necesario definir cuáles serán aquellos recursos que estarán disponibles en el servidor, y cómo estarán compuestos. Es decir, qué atributos tendrán, y de qué tipo. En particular, y dado que esos recursos viajarán indistintamente de un lado al otro entre el cliente y el servidor, la forma más usual de representar a dichos recursos será mediante su serialización en formato JSON. En el siguiente ejemplo, vemos un recurso “Reply” (respuesta), en su versión JSON, y una tabla detallando tipo de dato y descripción de cada parte del recurso.

Replies

A reply to a comment on a file.

```
{
  "kind": "drive#reply",
  "id": string,
  "createdTime": datetime,
  "modifiedTime": datetime,
  "author": {
    "kind": "drive#user",
    "displayName": string,
    "photoLink": string,
    "me": boolean,
    "permissionId": string,
    "emailAddress": string
  },
  "htmlContent": string,
  "content": string,
  "deleted": boolean,
  "action": string
}
```

Property name	Value	Description	Notes
`kind`	`string`	Identifies what kind of resource this is. Value: the fixed string `"drive#reply"`.	
`id`	`string`	The ID of the reply.	
`createdTime`	`datetime`	The time at which the reply was created (RFC 3339 date-time).	
`modifiedTime`	`datetime`	The last time the reply was modified (RFC 3339 date-time).	
`author`	`nested object`	The user who created the reply.	
`author.kind`	`string`	Identifies what kind of resource this is. Value: the fixed string `"drive#user"`.	
`author.displayName`	`string`	A plain text displayable name for this user.	
`author.photoLink`	`string`	A link to the user's profile photo, if available.	
`author.me`	`boolean`	Whether this user is the requesting user.	
`author.permissionId`	`string`	The user's ID as visible in Permission resources.	
`author.emailAddress`	`string`	The email address of the user. This may not be present in certain contexts if the user has not made their email address visible to the requester.	
`htmlContent`	`string`	The content of the reply with HTML formatting.	
`content`	`string`	The plain text content of the reply. This field is used for setting the content, while `htmlContent` should be displayed. This is required on creates if no `action` is specified.	writable
`deleted`	`boolean`	Whether the reply has been deleted. A deleted reply has no content.	
`action`	`string`	The action the reply performed to the parent comment. Valid values are: • `resolve` • `reopen`	writable

Nombre, ruta y descripción de cada método

Cada uno de los recursos presentes en el servidor contará con uno o más métodos relacionados, que nos permitirán interactuar con los mismos. Si bien cada método puede contar con un nombre descriptivo que lo identifique tanto en el código como en la documentación, es necesario vincular a cada uno de ellos con un método HTTP, y con la ruta a un recurso determinado, según corresponda. En este ejemplo se muestra el método “create” (crear) con relación al recurso “Reply” antes mencionado. Se incluye también una brevísima descripción del mismo.

Replies: create

☆☆☆☆☆

Creates a new reply to a comment. Try it now.

Acá se incluye la URI del recurso sobre el que se aplicará el método HTTP, en este caso post, que disparará el caso de uso de creación de un nuevo recurso en el servidor.

Request

HTTP request

```
POST https://www.googleapis.com/drive/v3/files/fileId/comments/commentId/replies
```

Como ven, la URI puede contener algunos parámetros en la ruta (en este caso, fileId y commentId), pero también podría contener algún parámetro luego de finalizar la ruta (los conocidos como *parámetros de consulta* o *query params*). En este caso vemos también para ambos tipos de parámetros, una tabla explicativa, definiendo tipo y descripcion de cada parámetro.

Parameters

Parameter name	Value	Description
Path parameters		
`commentId`	`string`	The ID of the comment.
`fileId`	`string`	The ID of the file.
Required query parameters		
`fields`	`string`	The paths of the fields you want included in the response. For development you can use the special value * to return all fields, but you'll achieve greater performance by only selecting the fields you need. For more information see the partial responses documentation.

Características de la petición y respuesta de cada método

Finalmente, se describe qué se envía junto con la petición, y qué devolverá el servidor junto con la respuesta al cliente. En este ejemplo, tanto la petición (post) como la respuesta, contienen en su cuerpo (atributo “body” de las peticiones) un recurso de tipo “Reply”, con algunos detalles añadidos, como indica la tabla a continuación.

Códigos de estado / errores

En esta sección se incluyen todos los posibles errores que puede devolver el servidor, y cómo se deben interpretar. En el ejemplo, se presenta una tabla con código de estado y descripción de la respuesta/error.

Errors

Mailgun returns standard HTTP response codes.

Code	Description
200	Everything worked as expected
400	Bad Request - Often missing a required parameter
401	Unauthorized - No valid API key provided
402	Request Failed - Parameters were valid but request failed
404	Not Found - The requested item doesn’t exist
413	Request Entity Too Large - Attachment size is too big
500, 502, 503, 504	Server Errors - something is wrong on Mailgun’s end

Fuentes:

https://developers.google.com/drive/api/v3/reference/replies (google drive)
https://documentation.mailgun.com/en/latest/api-intro.html#introduction (mailgun)